IN THE CLAIMS:

1. (Original) A host-fabric adapter, comprising:

one or more Micro-Engines arranged to establish connections and support data transfer operations, via a switched fabric, in response to work requests from a host system for data transfer operations; and

a remote key manager arranged to manage remote keys and check the validity of the remote keys which correspond to outstanding data transfer operations, via said switched fabric.

2. (Original) The host-fabric adapter as claimed in claim 1, further comprising a transport engine which contains a plurality of work queue pairs (WQPs) in which work requests in a form of descriptors are posted to describe data transfer operations and locations of data to be moved for processing and/or transportation via said switched fabric.

3. (Original) The host-fabric adapter as claimed in claim 2, wherein said work queue pairs (WQPs) each comprises:

a Send Queue (SQ) utilized as an "initiator" which requests normal message sends to remote Virtual Interfaces (VIs) of a remote system, remote direct memory access (RDMA) reads which request messages to be read from specific memory locations of said remote system, via said switched fabric, and remote direct memory access (RDMA) writes which request messages to be written onto specific memory locations of said remote system, via said switched fabric; and

a Receive Queue utilized as a "responder" which receives requests for messages from normal sends, RDMA reads and RDMA writes from said remote system, via said switched fabric.

4. (Currently Amended) The host-fabric adapter as claimed in claim 3, wherein said Micro-Engines and said remote key manager are configured ~~in accordance~~ to be compliant with the *"InfiniBand™ Specification"*, and implemented as part of an Application Specific Integrated Circuit (ASIC).

5. (Original) The host-fabric adapter as claimed in claim 3, wherein said Micro-Engines comprise:

a Send Queue (SQ) Micro-Engine arranged to control operations of the Send Queue (SQ) of the work queue pair (WQP); and

a Receive Queue (RQ) Micro-Engine arranged to control operations of the Receive Queue (RQ) of the work queue pair (WQP).

6. (Original) The host-fabric adapter as claimed in claim 5, wherein said remote key manager contains a remote key memory for storing stores remote keys utilized for comparison with remote keys included in work requests for key validation.

7. (Original) The host-fabric adapter as claimed in claim 6, wherein said remote keys are used to identify an appropriate page for virtual to physical address translation for outstanding data transfer operations including RDMA read/write operations.

8. (Original) The host-fabric adapter as claimed in claim 7, wherein said work requests are posted in a form of descriptors for normal data transfer operations, such as normal sends, RDMA writes and RDMA reads, or in a form of Bind descriptors for non data operations, such as moving or invalidating "Memory Windows" within particular memory regions of a host memory of said host system on the Send Queue (SQ) to restrict RDMA data transfer operations.

9. (Original) The host-fabric adapter as claimed in claim 6, wherein said remote key Memory is a random-access-memory (RAM) having a single read port and a single write port.

10. (Original) The host-fabric adapter as claimed in claim 8, wherein said remote key manager is activated to check the validity of remote keys by:

determining if a descriptor posted at the Send Queue (SQ) is a Bind descriptor for a non data operation;

if the descriptor posted is not a Bind Descriptor, processing for a normal data transfer operation, such as a normal send, a RDMA write and a RDMA read;

if the descriptor posted is a Bind Descriptor, determining if the "Memory Window" is invalidated;

if the "Memory Window" is not invalidated, noting that the "Memory Window" is moved and performing writes to the host memory of said host system to move the "Memory Window";

if the "Memory Window" is invalidated, performing writes to the host memory of said host system to destroy the "Memory Window";

searching through all outstanding RDMA operations which use the "Memory Window" to identify a remote key which used the "Memory Window";

if the remote key which used the "Memory Window" is identified, invalidating the remote key until all remote keys are invalidated;

if no remote key is identified, completing invalidation of all the remote keys and returning to being idle; and

if all remote keys are not invalidated, returning to search through all outstanding RDMA operations which uses the "Memory Window" until all remote keys are marked "invalidated" so that no new remote key or new work queue pair (WQP) can come in and use that "Memory Window".

11. (Currently Amended) ~~The~~ A host-fabric adapter, comprising: ~~as claimed in claim 10,~~

one or more Micro-Engines arranged to establish connections and support data transfer operations, via a switched fabric, in response to work requests from a host system for data transfer operations; and

a remote key manager arranged to manage remote keys and check the validity of the remote keys which correspond to outstanding data transfer operations, via said switched fabric;

wherein said work queue pairs (WQPs) each comprises:

a Send Queue (SQ) utilized as an "initiator" which requests normal message sends to remote Virtual Interfaces (VIs) of a remote system, remote direct memory access (RDMA) reads which request messages to be read from specific memory locations of said remote system, via said switched fabric, and remote direct memory access (RDMA) writes which request

messages to be written onto specific memory locations of said remote system, via said switched fabric; and

a Receive Queue utilized as a "responder" which receives requests for messages from normal sends, RDMA reads and RDMA writes from said remote system, via said switched fabric;

wherein said Micro-Engines comprise:

a Send Queue (SQ) Micro-Engine arranged to control operations of the Send Queue (SQ) of the work queue pair (WQP); and

a Receive Queue (RQ) Micro-Engine arranged to control operations of the Receive Queue (RQ) of the work queue pair (WQP);

wherein said remote key manager contains a remote key memory for storing stores remote keys utilized for comparison with remote keys included in work requests for key validation;

wherein said remote keys are used to identify an appropriate page for virtual to physical address translation for outstanding data transfer operations including RDMA read/write operations;

wherein said work requests are posted in a form of descriptors for normal data transfer operations, such as normal sends, RDMA writes and RDMA reads, or in a form of Bind descriptors for non data operations, such as moving or invalidating "Memory Windows" within particular memory regions of a host memory of said host system on the Send Queue (SQ) to restrict RDMA data transfer operations;

wherein said remote key manager is activated to check the validity of remote keys by:

determining if a descriptor posted at the Send Queue (SQ) is a Bind descriptor for a non data operation;

if the descriptor posted is not a Bind Descriptor, processing for a normal data transfer operation, such as a normal send, a RDMA write and a RDMA read;

if the descriptor posted is a Bind Descriptor, determining if the "Memory Window" is invalidated;

if the "Memory Window" is not invalidated, noting that the "Memory Window" is moved and performing writes to the host memory of said host system to move the "Memory Window";

if the "Memory Window" is invalidated, performing writes to the host memory of said host system to destroy the "Memory Window";

searching through all outstanding RDMA operations which use the "Memory Window" to identify a remote key which used the "Memory Window";

if the remote key which used the "Memory Window" is identified, invalidating the remote key until all remote keys are invalidated;

if no remote key is identified, completing invalidation of all the remote keys and returning to being idle; and

if all remote keys are not invalidated, returning to search through all outstanding RDMA operations which uses the "Memory Window" until all remote keys are marked "invalidated" so that no new remote key or new work queue pair (WQP) can come in and use that "Memory Window";

wherein said remote key manager is activated upon a request for key invalidation from said SQ Micro-Engine to invalidate remote keys by:

initiating a Protection Domain (PD) compare between the Memory PD and SQ supplied PD of a Virtual Interface (VI);

if the Memory PD does not match the SQ supplied PD, skipping processing the remote keys for that VI, and determining whether all VIs are done; and

if the Memory PD matches the SQ supplied PD, reading and initiating remote key compares sequentially for all remote keys associated with that VI for key invalidation.

12. (Original) The host-fabric adapter as claimed in claim 11, wherein said remote key manager initiates remote key compares for all remote keys associated with that VI by:

reading and comparing a first remote key from the remote key Memory with the SQ supplied key;

if the first remote key does not match the SQ supplied key, moving to check a next remote key;

if the first remote key matches the SQ supplied key, clearing a Valid bit corresponding to the first remote key in the remote key Memory;

reading and comparing a second remote key from the remote key Memory with the SQ supplied key;

if the second remote key does not match the SQ supplied key, moving to check the next remote key;

if the second remote key matches the SQ supplied key, clearing the Valid bit corresponding to the second remote key in the remote key Memory;

reading and comparing a third remote key from the remote key Memory with the SQ supplied key;

if the third memory remote key does not match the SQ supplied key, moving to check a next remote key;

if the third remote key matches the SQ supplied key, clearing the Valid bit corresponding to third remote key in the remote key Memory;

reading and comparing a last remote key from the remote key Memory with the SQ supplied key;

if the last memory remote key does not match the SQ supplied key, moving to determine whether all VIs are done; and

if the last remote key matches the SQ supplied key, clearing the Valid bit corresponding to the last remote key in the remote key Memory and determining whether all VIs are done.

13. (Original) A host-fabric adapter installed at a host system for connecting to a switched fabric of a data network, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers via said switched fabric;

a serial interface arranged to receive and transmit data packets from said switched fabric for data transfer operations;

a host interface arranged to receive and transmit work requests, in the form of descriptors, from said host system for data transfer operations;

a context memory arranged to store context information needed for said Micro-Engine (ME) to process work requests for data transfer operations;

a doorbell manager arranged to update the context information needed for said Micro-Engine (ME) to process work requests for data transfer operations; and

a remote key manager arranged to manage remote keys and check the validity of the remote keys which correspond to outstanding data transfer operations.

14. (Original) The host-fabric adapter as claimed in claim 13, further comprising a transport engine which contains a plurality of work queue pairs (WQPs) in which work requests in a form of descriptors are posted to describe data transfer operations and locations of data to be moved for processing and/or transportation via said switched fabric.

15. (Original) The host-fabric adapter as claimed in claim 14, wherein said work queue pairs (WQPs) each comprises:

a Send Queue (SQ) utilized as an "initiator" which requests normal message sends to remote Virtual Interfaces (VIs) of a remote system, remote direct memory access (RDMA) reads which request messages to be read from specific memory locations of said remote system, via said switched fabric, and remote direct memory access (RDMA) writes which request messages to be written onto specific memory locations of said remote system, via said switched fabric; and

a Receive Queue utilized as a "responder" which receives requests for messages from normal sends, RDMA reads and RDMA writes from said remote system, via said switched fabric.

16. (Currently Amended) The host-fabric adapter as claimed in claim 13, wherein said Micro-Engine, said host interface, said serial interface, said context memory, said doorbell manager, and said remote key manager are configured ~~in accordance~~ to be compliant with the

"InfiniBand™ Specification", and implemented as part of an Application Specific Integrated Circuit (ASIC).

17. (Original) The host-fabric adapter as claimed in claim 15, wherein said Micro-Engine includes:

a Send Queue (SQ) Micro-Engine arranged to control operations of the Send Queue (SQ) of the work queue pair (WQP); and

a Receive Queue (RQ) Micro-Engine arranged to control operations of the Receive Queue (RQ) of the work queue pair (WQP).

18. (Original) The host-fabric adapter as claimed in claim 17, wherein said remote key manager contains a remote key memory for storing stores remote keys utilized for comparison with remote keys included in work requests for key validation.

19. (Original) The host-fabric adapter as claimed in claim 18, wherein said remote keys are used to identify an appropriate page for virtual to physical address translation for outstanding data transfer operations including RDMA read/write operations.

20. (Original) The host-fabric adapter as claimed in claim 19, wherein said work requests are posted in a form of descriptors for normal data transfer operations, such as normal sends, RDMA writes and RDMA reads, or in a form of Bind descriptors for non data operations, such as moving or invalidating "Memory Windows" within particular memory regions of a host memory of said host system on the Send Queue (SQ) to restrict RDMA data transfer operations.

21. (Original) The host-fabric adapter as claimed in claim 18, wherein said remote key Memory is a 1280x28 random-access-memory (RAM) having a single read port and a single write port.

22. (Original) The host-fabric adapter as claimed in claim 20, wherein said remote key manager is activated to check the validity of remote keys by:

determining if a descriptor posted at the Send Queue (SQ) is a Bind descriptor for a non data operation;

if the descriptor posted is not a Bind Descriptor, processing for a normal data transfer operation, such as a normal send, a RDMA write and a RDMA read;

if the descriptor posted is a Bind Descriptor, determining if the "Memory Window" is invalidated;

if the "Memory Window" is not invalidated, noting that the "Memory Window" is moved and performing writes to the host memory of said host system to move the "Memory Window";

if the "Memory Window" is invalidated, performing writes to the host memory of said host system to destroy the "Memory Window";

searching through all outstanding RDMA operations which use the "Memory Window" to identify a remote key which used the "Memory Window";

if the remote key which used the "Memory Window" is identified, invalidating the remote key until all remote keys are invalidated;

if no remote key is identified, completing invalidation of all the remote keys and returning to being idle; and

if all remote keys are not invalidated, returning to search through all outstanding RDMA operations which uses the "Memory Window" until all remote keys are marked "invalidated" so that no new remote key or new work queue pair (WQP) can come in and use that "Memory Window".

23. (Original) The host-fabric adapter as claimed in claim 22, wherein said remote key manager is activated upon a request for key invalidation from said SQ Micro-Engine to invalidate remote keys by:

initiating a Protection Domain (PD) compare between the Memory PD and SQ supplied PD of a Virtual Interface (VI);

if the Memory PD does not match the SQ supplied PD, skipping processing the remote keys for that VI, and determining whether all VIs are done; and

if the Memory PD matches the SQ supplied PD, reading and initiating remote key compares sequentially for all remote keys associated with that VI for key invalidation.

24. (Original) The host-fabric adapter as claimed in claim 23, wherein said remote key manager initiates remote key compares for all remote keys associated with that VI by:

reading and comparing a first remote key from the remote key Memory with the SQ supplied key;

if the first remote key does not match the SQ supplied key, moving to check a next remote key;

if the first remote key matches the SQ supplied key, clearing a Valid bit corresponding to the first remote key in the remote key Memory;

reading and comparing a second remote key from the remote key Memory with the SQ supplied key;

if the second remote key does not match the SQ supplied key, moving to check the next remote key;

if the second remote key matches the SQ supplied key, clearing the Valid bit corresponding to the second remote key in the remote key Memory;

reading and comparing a third remote key from the remote key Memory with the SQ supplied key;

if the third memory remote key does not match the SQ supplied key, moving to check a next remote key;

if the third remote key matches the SQ supplied key, clearing the Valid bit corresponding to third remote key in the remote key Memory;

reading and comparing a last remote key from the remote key Memory with the SQ supplied key;

if the last memory remote key does not match the SQ supplied key, moving to determine whether all VIs are done; and

if the last remote key matches the SQ supplied key, clearing the Valid bit corresponding to the last remote key in the remote key Memory and determining whether all VIs are done.

25. (Original) A method of checking the validity of remote keys which correspond to outstanding remote direct memory access (RDMA) operations in a host-fabric adapter installed at a host system, comprising:

determining if any Virtual Interface (VI) to be processed at a Send Queue (SQ) is a Bind descriptor for a non data operation;

if the descriptor posted is not a Bind Descriptor, processing for a normal data transfer operation;

if the descriptor posted is a Bind Descriptor, determining if the "Memory Window" is invalidated;

if the "Memory Window" is not invalidated, noting that the "Memory Window" is moved and performing writes to a host memory of said host system to move the "Memory Window";

if the "Memory Window" is invalidated, performing writes to the host memory of said host system to destroy the "Memory Window";

searching through all outstanding RDMA operations which use the "Memory Window" to identify a remote key which used the "Memory Window";

if the remote key which used the "Memory Window" is identified, invalidating the remote key until all remote keys are invalidated;

if no remote key is identified, completing invalidation of all the remote keys and returning to being idle; and

if all remote keys are not invalidated, returning to search through all outstanding RDMA operations which uses the "Memory Window" until all remote keys are marked "invalidated" so that no new remote key or new work queue pair (WQP) can come in and use that "Memory Window".

26. (Original) The method as claimed in claim 25, wherein said remote keys are invalidated by:

initiating a Protection Domain (PD) compare between a Memory PD and SQ supplied PD of a Virtual Interface (VI);

if the Memory PD does not match the SQ supplied PD, skipping processing the remote keys for that VI, and determining whether all VIs are done; and

if the Memory PD matches the SQ supplied PD, reading and initiating remote key compares sequentially for all remote keys associated with that VI for key invalidation.

27. (Original) The method as claimed in claim 26, wherein said remote key compares for all remote keys associated with that VI are performed by:

reading and comparing a first remote key from a remote key Memory with the SQ supplied key;

if the first remote key does not match the SQ supplied key, moving to check a next remote key;

if the first remote key matches the SQ supplied key, clearing a Valid bit corresponding to the first remote key in the remote key Memory;

reading and comparing a second remote key from the remote key Memory with the SQ supplied key;

if the second remote key does not match the SQ supplied key, moving to check the next remote key;

if the second remote key matches the SQ supplied key, clearing the Valid bit corresponding to the second remote key in the remote key Memory;

reading and comparing a third remote key from the remote key Memory with the SQ supplied key;

if the third memory remote key does not match the SQ supplied key, moving to check a next remote key;

if the third remote key matches the SQ supplied key, clearing the Valid bit corresponding to third remote key in the remote key Memory;

reading and comparing a last remote key from the remote key Memory with the SQ supplied key;

if the last memory remote key does not match the SQ supplied key, moving to determine whether all VIs are done; and

if the last remote key matches the SQ supplied key, clearing the Valid bit corresponding to the last remote key in the remote key Memory and determining whether all VIs are done.

Please add new claims 28-36 as follows.

28. (New) A method for connecting a host-fabric adapter installed at a host system to a switched fabric of a data network, the method comprising:

establishing connections and supporting data transfers via said switched fabric;

receiving and transmitting data packets from said switched fabric for data transfer operations;

receiving and transmitting work requests, in the form of descriptors, from said host system for data transfer operations;

storing context information needed to process work requests for data transfer operations;

updating the context information needed to process work requests for data transfer operations; and

managing remote keys and checking the validity of the remote keys which correspond to outstanding data transfer operations.

29. (New) The method as claimed in claim 28, further comprising posting work requests in a form of descriptors to describe data transfer operations and locations of data to be moved for processing and/or transportation via said switched fabric.

30. (New) The method as claimed in claim 29, further comprising:

requesting normal message sends to remote Virtual Interfaces (VIs) of a remote system, requesting messages to be read from specific memory locations of said remote system, via said switched fabric, and requesting messages to be written onto specific memory locations of said remote system, via said switched fabric; and

receiving requests for messages from normal sends, reads and writes from said remote system, via said switched fabric.

31. (New) The method as claimed in claim 28, wherein the method is compliant with the *"InfiniBand™ Specification"*.

32. (New) The method as claimed in claim 28, further comprising storing remote keys utilized for comparison with remote keys included in work requests for key validation.

33. (New) The method as claimed in claim 32, further comprising identifying an appropriate page for virtual to physical address translation for outstanding data transfer operations.

34. (New) The method as claimed in claim 28, further comprising checking a validity of remote keys by:

determining if a descriptor posted at a Send Queue (SQ) is a Bind descriptor for a non data operation;

if the descriptor posted is not a Bind Descriptor, processing for a normal data transfer operation, such as a normal send, a RDMA write and a RDMA read;

if the descriptor posted is a Bind Descriptor, determining if a "Memory Window" is invalidated;

if the "Memory Window" is not invalidated, noting that the "Memory Window" is moved and performing writes to the host memory of said host system to move the "Memory Window";

if the "Memory Window" is invalidated, performing writes to the host memory of said host system to destroy the "Memory Window";

searching through all outstanding operations which use the "Memory Window" to identify a remote key which used the "Memory Window";

if the remote key which used the "Memory Window" is identified, invalidating the remote key until all remote keys are invalidated;

if no remote key is identified, completing invalidation of all the remote keys and returning to being idle; and

if all remote keys are not invalidated, returning to search through all outstanding operations which use the "Memory Window" until all remote keys are marked "invalidated" so that no new remote key or new work queue pair can come in and use that "Memory Window".

35. (New) The method as claimed in claim 34, further comprising activating the remote key manager upon a request for key invalidation to invalidate remote keys by:

initiating a Protection Domain (PD) compare between the Memory PD and SQ supplied PD of a Virtual Interface (VI);

if the Memory PD does not match the SQ supplied PD, skipping processing the remote keys for that VI, and determining whether all VIs are done; and

if the Memory PD matches the SQ supplied PD, reading and initiating remote key compares sequentially for all remote keys associated with that VI for key invalidation.

36. (New) The method as claimed in claim 35, wherein said remote key manager initiates remote key compares for all remote keys associated with that VI by:

reading and comparing a first remote key from the remote key Memory with the SQ supplied key;

if the first remote key does not match the SQ supplied key, moving to check a next remote key;

if the first remote key matches the SQ supplied key, clearing a Valid bit corresponding to the first remote key in the remote key Memory;

reading and comparing a second remote key from the remote key Memory with the SQ supplied key;

if the second remote key does not match the SQ supplied key, moving to check the next remote key;

if the second remote key matches the SQ supplied key, clearing the Valid bit corresponding to the second remote key in the remote key Memory;

reading and comparing a third remote key from the remote key Memory with the SQ supplied key;

if the third memory remote key does not match the SQ supplied key, moving to check a next remote key;

if the third remote key matches the SQ supplied key, clearing the Valid bit corresponding to third remote key in the remote key Memory;

reading and comparing a last remote key from the remote key Memory with the SQ supplied key;

if the last memory remote key does not match the SQ supplied key, moving to determine whether all VIs are done; and

if the last remote key matches the SQ supplied key, clearing the Valid bit corresponding to the last remote key in the remote key Memory and determining whether all VIs are done.